View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by IntechOpen

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

8

Adaptive Control Based On Neural Network

Sun Wei, Zhang Lujin, Zou Jinhai and Miao Siyi
College of Electrical and Information Engineering, Hunan University
Changsha City, Hunan Province, P. R. China

1. Introduction

Neural network has good nonlinear function approximation ability. It can be widely used to identify the model of controlled plant. In this chapter, the theories of modeling uncertain plant by using two kinds of neural networks: feed-forward neural network and recurrent neural network are introduced. And two adaptive control strategies for robotic tracking control are developed. One is recurrent fuzzy neural network based adaptive control (RFNNBAC), and another is neural network based adaptive robust control (NNBARC). In RFNNBAC, a kind of recurrent fuzzy neural network (RFNN) is constructed by using recurrent neural network to realize fuzzy inference, In which, temporal relations are embedded in the network by adding feedback connections on the first layer of the network. Two RFNNs are used to identify and control plant respectively. Base on the Lyapunov stability approach, the convergence of the proposed RFNN is analyzed. In NNBARC, A robust controller and a neural network are combined into an adaptive robust robotic tracking control scheme. Neural network is used to approximate the modeling uncertainties in robotic system. Then the disadvantageous effects on tracking performance, due to the approximating error of the neural network and non-measurable external disturbances in robotic system, are attenuated to a prescribed level by robust controller. The robust controller and the adaptation law of neural network are designed based on Hamilton-Jacobi-Issacs (HJI) inequality theorem. The weights of NN are easily tuned on-line by a simple adaptation law, with no need of a tedious and lengthy off-line training phase.
This chapter is organized in the following manner. In the first section a robust robotic tracking controller based on neural network is designed and its effectiveness is proved by applying it to control the trajectories of a two-link robot. Secondly, a recurrent fuzzy neural network based adaptive control is proposed and simulation experiments are made by applying it on robotic tracing control problem to confirm its effectiveness. Finally, some conclusions are drawn.

2. A robust robotic tracking controller based on neural network

In the past decades, there has been much research on the applications of nonlinear control theory to control robots, and many useful properties of robot dynamics such as the skew-symmetry property were discovered. There are basically two strategies to control such uncertain nonlinear systems: the robust control strategy and the adaptive control strategy. A

convenient point of robust control strategy is that it can attenuate disadvantageous effects of various uncertainties (e.g., structured parametric uncertainties and unstructured disturbances) to a required level, provided that the upper bound of uncertainties is well known (Abdallah et al. 1991). However, since this strategy use max-min method to design the controller, it can not yield good transient performance. On the other hand, regressor matrixes are always used in the design of adaptive control systems for robot manipulators (Ortega & Spong 1989). In this situation, the unknown nonlinear dynamics of robotic systems are always assumed to be linearly parametrisable. However, there are some potential difficulties associated with this classical adaptive control design. For example, the unknown parameters may be quickly varying, the linear parametrisable property may not hold, computation of the regressor matrix is a time-consuming task, and implementation also requires a precise knowledge of the structure of the entire robot dynamic model (Saad et al. 1994; Sanner & Slotine 1998; Spooner & Passino 1996).

It has been shown that multi-layer neural networks can approximate any continuous function as accurately as possible. Based on this universal approximation property, many important adaptive neural-network-based control schemes have been developed to solve highly nonlinear control problem (Sanner & Slotine 1998; Spooner & Passino 1996; Narenra & Parthasarathy 1990; Polycarpou 1996). But most of these schemes use grads-descent method to train the weights, which can not ensure the stability of whole closed-loop system. In the recent years, researchers began to develop the neural-network-based controller with closed-loop stability based on the Lyapunov method. A controller based on forward propagation network was developed in (Carelli et al. 1995), but it didn't consider the effects of uncertainties. An adaptive neural network control strategy with guaranteed stability was proposed in (Behera et al. 1996) on the assumption that the approximation error of the neural network is known and bounded.

In the first part of this chapter, we will propose a neural-network-based robust robotic tracking controller according to HJI inequation theorem presented by Shen in (Shen 1996). A neural network equipped with a robust learning algorithm is introduced firstly to learn the modeling uncertainties in robotic system. Then the disadvantageous effects on tracking performance caused by neural network approximating error and non-measurable external disturbances in robotic system will be attenuated to a prescribed level by the designing a robust controller.

This section is organized as follows. In subsection 2.1, HJI inequation theorem is introduced. In subsection 2.2 the dynamics of robot system and its properties are described. The neural network based robust control strategy is proposed in subsection 2.3, where the structure of robust controller and the robust learning algorithm of neural network are derived. Simulations for a two-link robot are presented in subsection 2.4.

2.1 HJI inequation theorem

A system with non-measurable disturbance d can be formulated as:

$$\dot{x} = f(x) + g(x)d \tag{1}$$

For evaluating the disturbance restraint performance of system (1), an evaluation signal $z = h(x)$ is introduced to represent the signals need to be concerned, such as error. And a

performance index signal can be defined as:

$$J = \sup_{\|d\|_2 \neq 0} \frac{\|z\|_2}{\|d\|_2} \tag{2}$$

Obviously, smaller J means better disturbance restraint performance. The robust design problem of system (1) can be solved by designing a controller to make J less than a prescribed level.

HJI(Hamilton-Jacobi-Isaacs)InequationTheorem: Given an positive constant $\gamma > 0$, if there exists an derivable function, V(x)≥0, which satisfies the following HJI inequation:

$$\dot{V} = \frac{\partial V}{\partial x}\dot{x} = \frac{\partial V}{\partial x}f(x) + \frac{\partial V}{\partial x}g(x)d \leq \frac{1}{2}\left\{\gamma^2\|d\|^2 - \|z\|^2\right\}, \forall d \tag{3}$$

then the performance index signal of system (1) is less than γ, that is to say, $J \leq \gamma$.

2.2 Problem statement

The kinetics equation of a robotic manipulator with uncertainties can be expressed as:

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} + G(q) + \Delta T(q,\dot{q}) + d_R = T \tag{4}$$

where $q, \dot{q}, \ddot{q} \in R^n$ is the joint position, velocity, and acceleration vectors; $M(q) \in R^{n\times n}$ denotes the moment of inertia; $V(q,\dot{q})\dot{q}$ are the Coriolis and centripetal forces; $G(q)$ includes the gravitational forces; T is the applied torque; $\Delta T(q,\dot{q})$ represents the modelling uncertainties in robotic system, and d_R is external non-measurable disturbance.

It is well known that the robot dynamics has the following properties.

Property 1— Boundedness of the Inertia matrix: The inertia matrix $M(q)$ is symmetric and positive definite, and satisfies the following inequalities:

$$0 < \lambda_m I \leq M(q) \leq \lambda_M I \tag{5}$$

where λ_m and λ_M are known positive constants.

Property 2—Skew symmetry: The inertia and centripetal-Coriolis matrices have the following property:

$$\xi^T\{\dot{M}(q) - 2V(q,\dot{q})\}\xi = 0, \forall \xi \in R^n \tag{6}$$

Property 1 is very important in generating a positive definite function to prove the stability of the closed-loop system. Property 2 will help in simplifying the controller.

The aim of this paper is to design a neural-network-based robust controller (NNBRC) for the robot system under uncertainties, such that closed-loop system is guaranteed to be stable

and the joint position $q(t)$ can track the desired trajectory $q_d(t)$ rapidly and accurately.

2.3 Design of NNBRC

A NNBRC is proposed in this section. In the proposed strategy, a neural network (NN) is firstly used for identifing modelling uncertainties $\Delta T(q,\dot{q})$, then, a robust learning algorithm and a robust controller are designed based on HJI equation theorem to counteract the disadvantageous effects caused by approximation error of the NN and external disturbance d_R .

2.3.1 Construction of the neural network

A three-layer NN is shown in Fig.1.Using $u_i^{(1)}, o_i^{(1)}$ to denote the input and output of the ith node in the lth layer separately, the signal propagation and the operation functions of the nodes in each layer are introduced as follows.

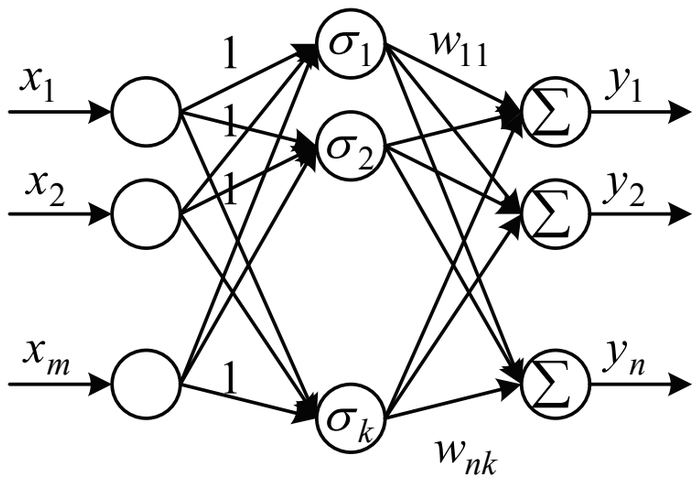

Fig. 1. Structure of three-layer NN

Layer 1— Input Layer:

$$o_i^{(1)} = u_i^{(1)} = x_i, i = 1,2,\cdots,m \tag{7}$$

Layer 2— Hidden Layer:

$$u_j^{(2)} = \sum_{i=1}^{m} o_i^{(1)}, \ j = 1,2,\cdots,k \tag{8}$$

$$o_j^{(2)} = \sigma_j = 1/[1+\exp(-u_j^{(2)})], j = 1,2,\cdots,k \tag{9}$$

Layer 3 – Output Layer:

$$y_h = o_h^{(3)} = u_h^{(3)} = \sum_{j=1}^{k} w_{hj} \cdot o_j^{(2)}, h = 1,2,\cdots,n \tag{10}$$

Let

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1k} \\ w_{21} & w_{22} & \cdots & w_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nk} \end{bmatrix}, \sigma = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_k \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

then the outputs of the three-layer NN can be written as:

$$Y = W\sigma \tag{11}$$

In this paper, the three-layer NN described above will be used to identify the modeling uncertainties $\Delta T(q,\dot{q})$ in robotic system. Using ε_T to denote the network approximation error, then the modeling uncertainties can be denoted by:

$$\Delta T\ (q,\dot{q}) = W_T\sigma_T + \varepsilon_T \Delta T(q,\dot{q}) \tag{12}$$

where W_T is the weight matrix, σ_T is the activation function vector.
Substitute (12) into (4), then the dynamics of the robot manipulator with a NN identifier can be formulated as:

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} + G(q) + W_T\sigma_T + \varepsilon_T + d_R = T \tag{13}$$

Regarding ε_T as another external disturbance of robotic system, and using $\varepsilon_R = \varepsilon_T + d_R$, then (13) can be rewritten as:

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} + G(q) + W_T\sigma_T + \varepsilon_R = T \tag{14}$$

For attenuating disadvantageous effects caused by ε_R to a prescribed level, a robust learning algorithm of NN and a robust controller can be designed based on HJI equation as below 2.3.2.

2.3.2 Robust controller and NN learning algorithm

At first, we introduce a control signal u, which satisfies:

$$M(q)\ddot{q}_d + V(q,\dot{q})\dot{q}_d + G(q) + u = T \tag{15}$$

where $q_d, \dot{q}_d, \ddot{q}_d \in R^n$ is desired joint position, velocity, and acceleration vectors separately.

Thus, the closed-loop robot control system can be constructed by substituting (15) into (14). Let $e = q - q_d$, the closed-loop system can be formulated as:

$$M(q)\ddot{e} + V(q,\dot{q})\dot{e} + W_T\sigma_T + \varepsilon_R = u \tag{16}$$

By regarding ε_R as external disturbance and introducing the evaluation signal $z_R = pe$, where p is a positive constant, we can define the index signal as:

$$J_R = \sup_{\|\varepsilon_R\|_2 \neq 0} \frac{\|z_R\|_2}{\|\varepsilon_R\|_2} \tag{17}$$

The idea of NNBRC is to design controller u and the NN learning algorithm $\dot{W}_T$ such that J_R is less than a prescribed level, γ.
Define two state variables as:

$$\begin{cases} x_1 = e \\ x_2 = \dot{e} + \alpha e \end{cases} \tag{18}$$

where α is an prescribed positive constant. Thus, system (16) can be rewritten as:

$$\begin{cases} \dot{x}_1 = x_2 - \alpha x_1 \\ M\dot{x}_2 = -Vx_2 + \omega - W_T\sigma_T - \varepsilon_R + u \end{cases} \tag{19}$$

where $\omega = M\alpha\dot{e} + V\alpha e$, W_T is a $n \times k$ matrix that can be described as:

$$W_T = \begin{bmatrix} w_{T11} & w_{T12} & \cdots & w_{T1k} \\ w_{T21} & w_{T22} & \cdots & w_{T2k} \\ \vdots & \vdots & \vdots & \vdots \\ w_{Tn1} & w_{Tn2} & \cdots & w_{Tnk} \end{bmatrix} = \begin{bmatrix} w_{T1} & w_{T2} & \cdots & w_{Tk} \end{bmatrix}$$

Theorem 1: Considering system (19), if the learning algorithm of NN is:

$$\dot{W}_T = -\eta W_T \tag{20}$$

The controller u is designed as:

$$u = -x_1 - \omega + W_T\sigma_T - \left(\varepsilon_2 + \frac{1}{2Y^2}\right)x_2 \tag{21}$$

and the parameter p in the *evaluation signal* , $z_R = pe = px_1$, satisfies:

$$\alpha - \frac{1}{2}p^2 = \varepsilon_1 \tag{22}$$

where $\varepsilon_1, \varepsilon_2$ and η are all prescribed positive constant, then the disturbance restraint index signal of system (19), J_R, is less than γ.

Proof: Considering system (19), we define the following derivable function:

$$L = \frac{1}{2}\cdot x_1^T \cdot x_1 + \frac{1}{2}\cdot x_2^T \cdot M \cdot x_2 + \frac{1}{2}\cdot \|W_T\|^2 \tag{23}$$

Thus,

$$\begin{aligned}\dot{L} &= x_1^T \dot{x}_1 + x_2^T M \dot{x}_2 + \frac{1}{2}x_2^T \dot{M} x_2 + \sum_{i=1}^{m} w_{Ti}^T \dot{w}_{Ti} \\ &= x_1^T (x_2 - \alpha x_1) + x_2^T (\omega - W_T \sigma_T - \varepsilon_R + u) \\ &\quad + \frac{1}{2}x_2^T (\dot{M} - 2V) x_2 + \sum_{i=1}^{m} w_{Ti}^T \dot{w}_{Ti}\end{aligned}$$

According to Property 2 of the robot dynamics, the above equation can be rewritten as:

$$\begin{aligned}\dot{L} &= x_1^T (x_2 - \alpha x_1) + x_2^T (\omega - w_T \sigma_T - \varepsilon_R + u) + \sum_{i=1}^{m} w_{Ti}^T \dot{w}_{Ti} \\ &= -x_1^T \alpha x_1 + x_2^T (x_1 + \omega - w_T \sigma_T - \varepsilon_R + u) + \sum_{i=1}^{m} w_{Ti}^T \dot{w}_{Ti}\end{aligned}$$

Substituting (20) into above equation, then

$$\dot{L} = -x_1^T \alpha x_1 + x_2^T (x_1 + \omega - W_T \sigma_T - \varepsilon_R + u) - \eta \|W_T\|^2$$

Regarding ε_R as external disturbance, let

$$\begin{aligned}H &= \dot{L} - \frac{1}{2}\gamma^2 \|\varepsilon_R\|^2 + \frac{1}{2}\|z_R\|^2 \\ &= -\alpha \|x_1\|^2 + x_2^T (x_1 + \omega - W_T \sigma_T + u) - x_2^T \varepsilon_R \\ &\quad - \eta \|W_T\|^2 - \frac{1}{2}\gamma^2 \|\varepsilon_R\|^2 + \frac{1}{2}p^2 \|x_1\|^2 \\ &= -\left(\alpha - \frac{1}{2}p^2\right)\|x_1\|^2 + x_2^T (x_1 + \omega - W_T \sigma_T + u) \\ &\quad - x_2^T \varepsilon_R - \eta \|W_T\|^2 - \frac{1}{2}\gamma^2 \|\varepsilon_R\|^2\end{aligned}$$

$$\because -2x_2^T\varepsilon_R - \gamma^2\|\varepsilon_R\|^2 = -\left\{2x_2^T\varepsilon_R + \gamma^2\|\varepsilon_R\|^2\right\}$$

$$= -\left\{\frac{1}{\gamma^2}\|x_2\|^2 + 2x_2^T\varepsilon_R + \gamma^2\|\varepsilon_R\|^2 - \frac{1}{\gamma^2}\|x_2\|^2\right\}$$

$$= -\left\{\left\|\frac{1}{\gamma}x_2 + \gamma\varepsilon_R\right\|^2 - \frac{1}{\gamma^2}\|x_2\|^2\right\}$$

$$= -\left\|\frac{1}{\gamma}x_2 + \gamma\varepsilon_R\right\|^2 + \frac{1}{\gamma^2}\|x_2\|^2$$

$$\le \frac{1}{\gamma^2}\|x_2\|^2$$

$$\therefore -x_2^T\varepsilon_R - \frac{1}{2}\gamma^2\|\varepsilon_R\|^2 \le \frac{1}{2\gamma^2}\|x_2\|^2$$

$$H \le -\left(\alpha - \frac{1}{2}p^2\right)\|x_1\|^2 + x_2^T(x_1 + \omega - W_T\sigma_T + u) - \eta\|W_T\|^2 + \frac{1}{2\gamma^2}\|x_2\|^2$$

$$= -\left(\alpha - \frac{1}{2}p^2\right)\|x_1\|^2 + x_2^T\left(x_1 + \omega - W_T\sigma_T + u + \frac{1}{2\gamma^2}x_2\right) - \eta\|W_T\|^2$$

Substituting (21), (22) into above inequation, then

$$H \le -\varepsilon_1\|x_1\|^2 - \varepsilon_2\|x_2\|^2 - \eta\|W_T\|^2 \le 0$$

$$\therefore \dot{L} \le \frac{1}{2}\left\{\gamma^2\|\varepsilon_R\|^2 - \|z_R\|^2\right\}$$

According to HJI equation theorem, we can conclude that the disturbance restraint performance index signal of system (19), J_R, is less than γ. The structure of the proposed neural network based robust control strategy is illustrated in Fig. 2.

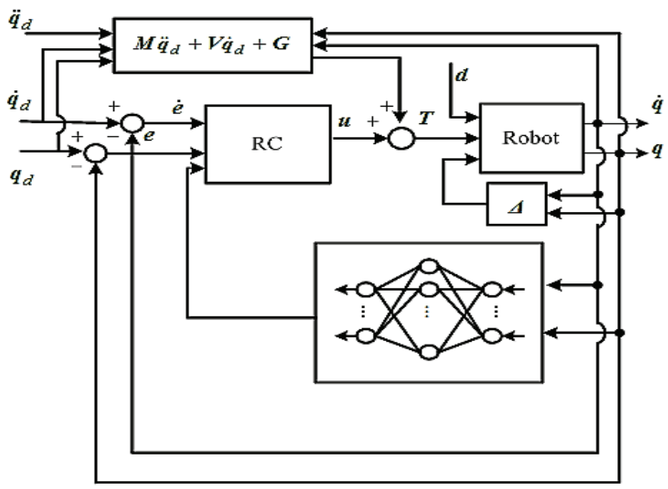

Fig. 2. Structure of the NN-based robust tracking control system

2.4 Simulation example

In this section, the proposed control strategy will be applied to control the trajectory of a two-link robot (see Fig. 3) for proving its effectiveness.

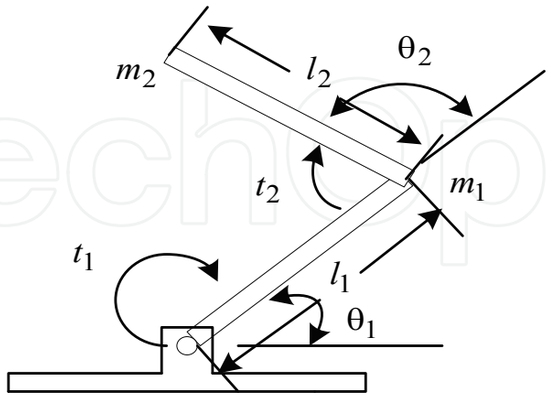

Fig. 3. Two-link robot

In Fig.3, m1and m2 are masses of arm1 and arm2 respectively; l1 and l2 are lengths of arm1 and arm2; t1 and t2 are torques on arm1 and arm2; θ_1 and θ_2 are positions of arm1 and arm2. The dynamics model of two-link robot is same as (4).
Let

$$q = [\theta_1 \quad \theta_2]^T,\ \dot{q} = [\dot{\theta}_1 \quad \dot{\theta}_2]^T \tag{24}$$

$$\ddot{q} = [\ddot{\theta}_1 \quad \ddot{\theta}_2]^T,\ T = [t_1 \quad t_2]^T \tag{25}$$

$$c_i \equiv \cos\theta_i,\ s_i \equiv \sin\theta_i,\ c_{ij} \equiv \cos(\theta_i + \theta_j) \tag{26}$$

then M, V,G in (4) can de described as:

$$M(q) = \begin{bmatrix} m_1 l_1^2 + m_2(l_1^2 + l_2^2 + 2l_1 l_2 c_2) & m_2 l_2^2 + m_2 l_1 l_2 c_2 \\ m_2 l_2^2 + m_2 l_1 l_2 c_2 & m_2 l_2^2 \end{bmatrix} \tag{27}$$

$$V(q,\dot{q}) = \begin{bmatrix} -2m_2 l_1 l_2 s_2 \dot{q}_2 & -m_1 l_1 l_2 s_2 \dot{q}_2 \\ m_2 l_1 l_2 s_2 \dot{q}_1 & 0 \end{bmatrix} \tag{28}$$

$$G(q) = \begin{bmatrix} m_2 l_2 g c_{12} + (m_1 + m_2) l_1 g c_1 \\ m_2 l_2 g c_{12} \end{bmatrix} \tag{29}$$

In this paper, the parameters of the two-link robot are $m_1 = 10$ kg, $m_2 = 2$ kg, $l_1 = 1.1$ m, and $l_2 = 0.8$ m. The Initial states are $q(0) = [0.5 \quad 0.5]$ rad, $\dot{q}(0) = [0 \quad 0]^T$ rad/s, and $\ddot{q}(0) = [0 \quad 0]^T$ rad/s². The desired trajectories can be described as:

$$q_d(t) = [\sin(2\pi t) \quad \cos(2\pi t)]^T \text{ rad} \tag{30}$$

$$\dot{q}_d(t) = [2\pi\cos(2\pi t) \quad -2\pi\sin(2\pi t)]^T \text{ rad/s} \tag{31}$$

$$\ddot{q}_d(t) = [-4\pi^2\sin(2\pi t) \quad -4\pi^2\cos(2\pi t)]^T \text{ rad/s}^2 \tag{32}$$

The model error due to friction is assumed as:

$$\Delta T = \begin{bmatrix} 0.5\text{sign}(\dot{e}_1)[0.1 + \exp(-|\dot{e}_1|)] \\ \text{sign}(\dot{e}_2)[0.2 + \exp(-|\dot{e}_2|)] \end{bmatrix} \text{ N·m} \tag{33}$$

The external disturbance, $d_R = [d_1 \quad d_2]^T$ is a random signal which amplitude is less than 10N·m. In simulations, the NNBRC can be designed based on (21), in which $\alpha = 50$, $\varepsilon_1 = 0.1$, $\varepsilon_2 = 0.1$, $\gamma = 0.05$, $p = 9$. The NN learning algorithm is designed according to (20), where $\eta = 0.1$.

Fig.4 and Fig.5 present the simulation experiment results, in which, proposed control strategy is compared to traditional robust control (TRC) strategy. From these results, we can conclude that the NN-based robust tracking control strategy proposed in this paper can counteract disadvantageous effects caused by uncertainties in robotic system efficiently, and can achieve better transient performance than traditional robust control.

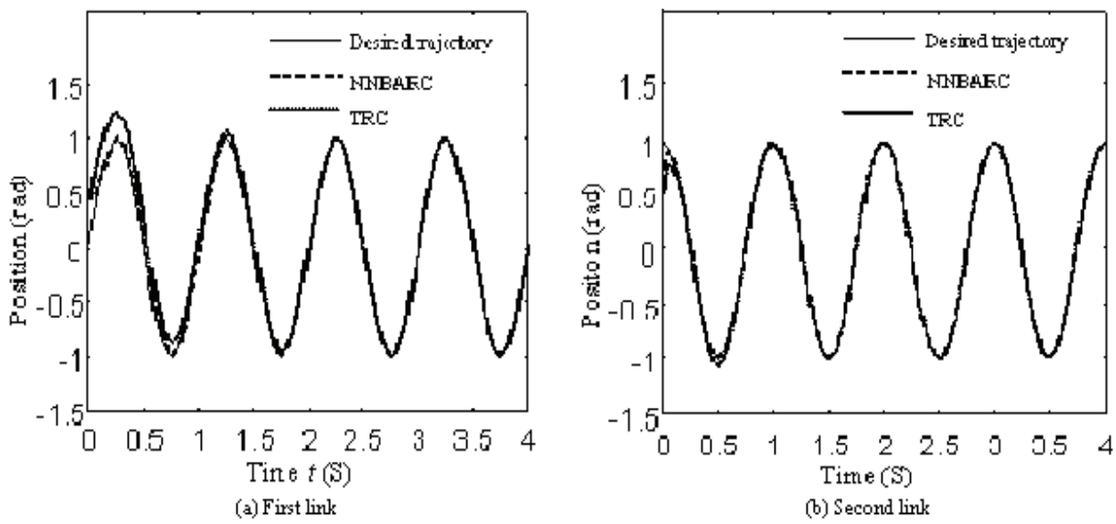

Fig. 4. Robot trajectories

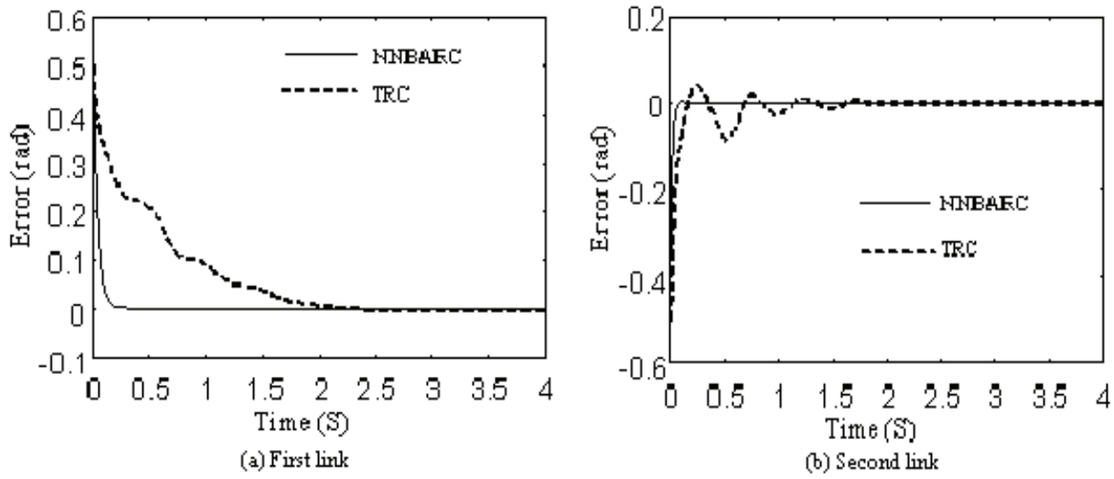

Fig. 5. Robot tracking errors

3. A Recurrent Fuzzy Neural Network Based Adaptive Control

Recently, much research has been done on using neural networks (NN) to identify and control dynamic systems (Park et al. 1996; Narendra & Parthasarathy 1990; Brdys & Kulawski 1999). NN can be classified as feed forward neural networks and recurrent neural networks. Feed forward neural networks can approximate a continuous function to an arbitrary degree of accuracy. However, feed forward neural network is a static mapping; it can not represent a dynamic mapping. Although this problem can be solved by using tapped delays, feed forward neural network requires a large number of neurons to represent dynamical responses in the time domain. Moreover, since the weight updates of feed forward neural network is irrelative to the internal information of neural network, the function approximation is sensitive to the training data. On the other hand, recurrent neural networks (Ku & Lee 1995; Ma & Ji 1998; Sundareshan & Condarcure 1998; Liang & Wang 2000) are able to represent dynamic mapping very well and store the internal information for updating weights later. Recurrent neural network has an internal feedback loop; it captures the dynamical response of a system without external feedback through delays. Recurrent neural network is a dynamic mapping and demonstrates good performance in the presence of uncertainties, such as parameter variations, external disturbance, unmodeled and nonlinear dynamics. However, the drawbacks of recurrent neural network, which are same as neural network, are that the function of the network is difficult to interpret and few efficient constructive methods can be found for choosing network structure and determining the parameters of neurons.

As is widely known, both fuzzy logic systems and neural network systems are aimed at exploiting human-like knowledge processing capability. In recent years, researchers started to recognize that fuzzy control has some similarities to neural network (Jang & Sun 1993; Hunt et al. 1996; Buckley et al. 1993; Reyneri 1999). Fuzzy neural network (FNN), which uses NN to realize fuzzy inference, combines the capability of fuzzy reasoning in handling uncertain information and the capability of neural networks in learning from processes. It is

possible to train NN using the experience of human operators expressed in term of linguistic rules, and interpret the knowledge that NN acquired from training data in linguistic form. And it is very easy to choose the structure of NN and determine the parameters of neurons from linguistic rules. However, a major drawback of the FNN is that its application domain is limited to static problems due to its feed forward network structure.

Recurrent fuzzy neural network (RFNN) is a modified version of FNN, which use recurrent network for realizing fuzzy inference and can be constructed from a set of fuzzy rules. It inherits all characteristics of FNN such as fuzzy inference, universal approximation and convergence properties. Moreover, with its own internal feedback connections, RFNN can temporarily store dynamic information and cope with temporal problems efficiently. For this ability to temporarily store information, the structure of RFNN is much simpler than FNN. Fewer nodes are required in RFNN for system identification.

In this section, a recurrent fuzzy neural network structure is proposed, in which, the temporal relations are embedded by adding feedback connections on the first layer of FNN. Back propagation algorithm is used to train the proposed RFNN. To guarantee the convergence of the RFNN, the Lyapunov stability approach is applied to select appropriate learning rates. For control problem, an adaptive control scheme is proposed, in which, two proposed RFNN are used to identify and control plant respectively. Finally, simulation experiments are made by applying proposed adaptive control scheme on robotic tracking control problem to confirm its effectiveness.

This section is organized as follows. In subsection 3.2, RFNN is constructed. The construction of RFNNBAC is presented in subsection 3.3. Learning algorithms of RFNN are derived in subsection 3.4. Stability of RFNN is analyzed in subsection 3.5. In subsection 3.6 proposed RFNNBAC is applied on robotic tracking control and simulation results are given. Finally, some conclusions are drawn in subsection 3.7.

3.1 Construction of RFNN

The structure of the proposed RFNN is shown in Fig. 6, which comprises n input variables, m term nodes for each input variable, l rule nodes, and p output nodes. This RFNN thus consists of four layers and $n + (n \times m) + l + p$ nodes.

Using u_i^k , O_i^k to denote the input and output of the ith node in the kth layer separately, the signal propagation and the operation functions of the nodes in each layer are introduced as follows.

Layer 1 (Input Layer): This layer accepts input variables. Its nodes transmit input values to the next layer. Feedback connections are added in this layer to embed temporal relations in the network. For every node in this layer, the input and output are represented as:

$$u_i^1(k) = x_i^1(k) + w_i^1 O_i^1(k-1), O_i^1(k) = u_i^1(k), i = 1,2,\cdots,n \tag{34}$$

where k is the number of iterations; w_i^1 is the recurrent weights.

Layer 2 (Membership Layer): Nodes in this layer represent the terms of respective linguistic variables. Each node performs a Gaussian membership function

$$u_{ij}^2 = -\frac{(O_i^1 - a_{ij})^2}{(b_{ij})^2}, O_{ij}^2 = \exp(u_{ij}^2) \tag{35}$$

where $i = 1,2,\cdots,n$, $j = 1,2,\cdots,m$; a_{ij} and b_{ij} are the mean and the standard deviation of the Gaussian membership function; the subscript ij indicates the jth term of the ith input variable.

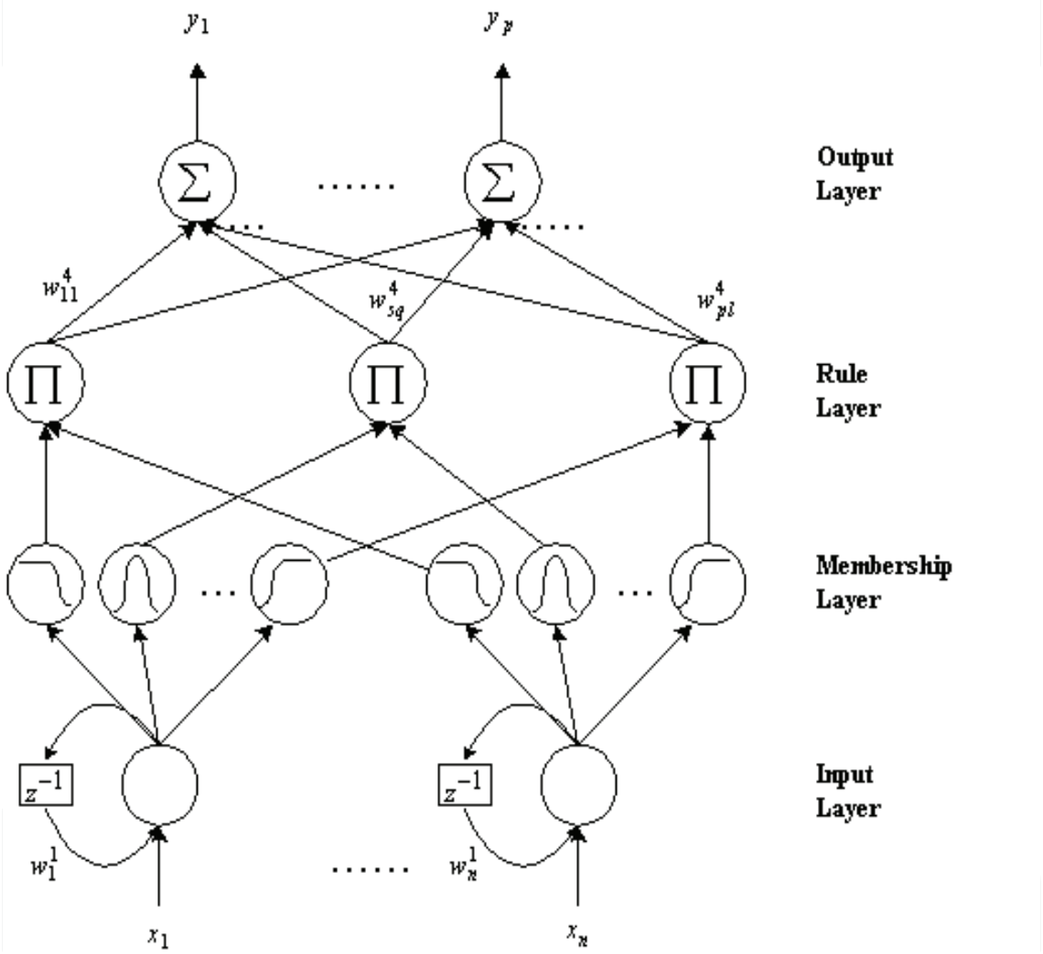

Fig. 6. Structure of four-layer RFNN

Layer 3(Rule Layer): This layer forms the fuzzy rule base and realizes the fuzzy inference. Each node is corresponding to a fuzzy rule. Links before each node represent the preconditions of the corresponding rule, and the node output represents the "firing strength" of corresponding rule.
If the qth fuzzy rule can be described as:

qth rule: if x_1 is A_1^q, x_2 is A_2^q, … , x_n is A_n^q then y_1 is B_1^q, y_2 is B_2^q, … , y_p is B_p^q, where A_i^q is the term of the ith input in the qth rule; B_j^q is the term of the jth output in the qth rule.

Then, the qth node of layer 3 performs the AND operation in qth rule. It multiplies the input signals and output the product.

Using $O_{iq_i}^2$ to denote the membership of x_i to A_i^q, where $q_i \in \{1,2,\cdots,m\}$, then the input and output of qth node can be described as:

$$u_q^3 = \prod_i O_{iq_i}^2 \ , O_q^3 = u_q^3, i = 1,2,\cdots,n; q = 1,2,\cdots,l \tag{36}$$

Layer 4(Output Layer): Nodes in this layer performs the defuzzification operation. the input and output of sth node can be calculated by:

$$u_s^4 = \sum_q w_{sq}^4 O_q^3 \ , \ O_s^4 = \frac{u_s^4}{\sum_q O_q^3} \tag{37}$$

where $s = 1,2,\cdots,p$, $q = 1,2,\cdots,l$, w_{sq}^4 is the center of B_j^q , which represents the output action strength of the sth output associated with the qth rule.

From the above description, it is clear that the proposed RFNN is a fuzzy logic system with memory elements in first layer. The RFNN features dynamic mapping with feedback and more tuning parameters than the FNN. In the above formulas, if the weights in the feedback unit w_i^1 are all equal to zero, then the RFNN reduces to an FNN. Since a fuzzy system has clear physical meaning, it is very easy to choose the number of nodes in each layer of RFNN and determine the initial value of weights. Note that the parameters w_i^1 of the feedback units are not set from human knowledge. According to the requirements of the system, they will be given proper values representing the memorized information. Usually the initial values of them are set to zero.

3.2 Structure of RFNNBAC

In this section, the structure of RFNNBAC will be developed below, in which, two proposed RFNN are used to identify and control plant respectively.

3.2.1 Identification based on RFNN

Resume that a system to be identified can be modeled by an equation of the following form:

$$y(k) = f\left(y(k-1),\cdots y\left(k-n_y\right), u(k),\cdots,u\left(k-n_u\right)\right) \tag{38}$$

where u is the input of the system, n_y is the delay of the output, and n_u is the delay of the input.

Feed forward neural network can be applied to identify above system by using y(k-1),… ,y(k-n-1), u(k), … , u(k-m) as inputs and approximating the function f.

For RFNN, the overall representation of inputs x and the output y can be formulated as

$$y(k) = g(O_1^1(k), \cdots, O_n^1(k)) \tag{39}$$

Where

$$\begin{aligned} O_i^1(k) &= x_i(k) + w_i^1(k) O_i^1(k-1) \\ &= x_i(k) + w_i^1(k)\left[x_i(k-1) + w_i^1(k-1) O_i^1(k-2)\right] \\ &\quad \vdots \\ &= x_i(k) + w_i^1(k) x_i(k-1) + w_i^1(k) w_i^1(k-1) x_i(k-2) + \cdots \\ &\quad + w_i^1(k) w_i^1(k-1) \cdots w_i^1(1) x_i(0) \end{aligned}$$

Using the current input u(k) and the most recent output y(k-1) of the system as the inputs of RFNN, (39) can be modified as:

$$\hat{y}(k) = \hat{f}(y(k-1), \cdots, y(0), u(k), \cdots, u(0)) \tag{40}$$

By training the RFNN according to the error e(k) between the actual system output and the RFNN output, the RFNN will estimate the output trajectories of the nonlinear system (38). The training model is shown in Fig.7.

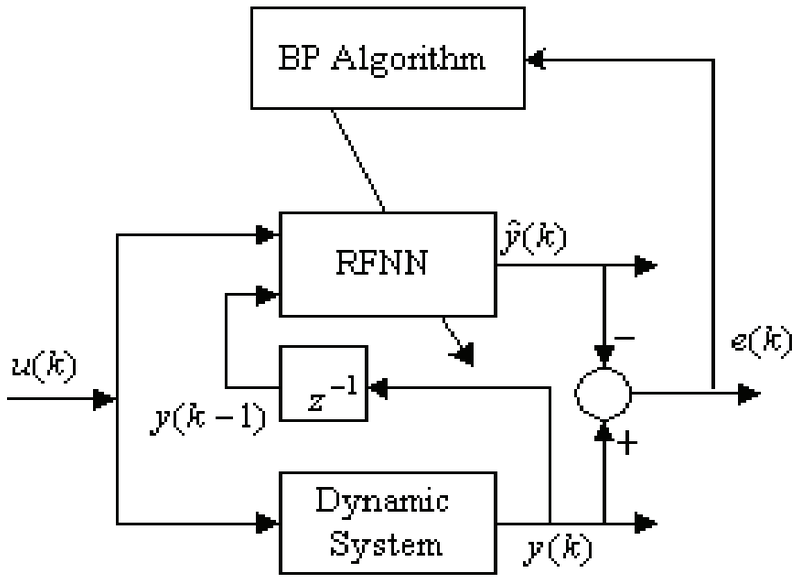

Fig. 7. Identification of dynamic system using RFNN

From above description, For Using RFNN to identify nonlinear system, only y(k-1) and u(k) need to be fed into the network .This simplifies the network structure, i. e., reduces the number of neurons

3.2.2 RFNNBAC

The block diagram of RFNNBAC is shown in Fig. 8. In this scheme, two RFNNs are used as controller (RFNNC) and identifier (RFNNI) separately. The plant is identified by RFNNI, which provides the information about the plant to RFNNC. The inputs of RFNNC are e(k) and $\dot{e}(k)$. e(k) is the error between the desired output r(t) and the actual system output y(k). The output of RFNNC is the control signal u(k), which drives the plant such that e(k) is minimized. In the proposed system, both RFNNC and RFNNI have same structure.

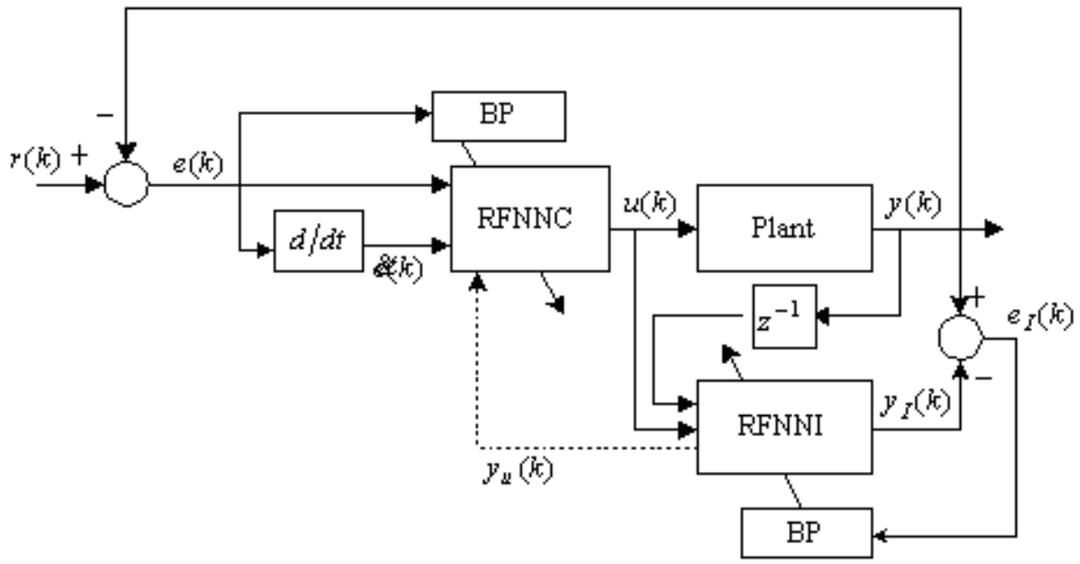

Fig. 8. Control system based on RFNNs

3.3 Learning Algorithm of RFNN

For parameter learning, we will develop a recursive learning algorithm based on the back propagation method

3.3.1 Learning algorithm for identifier

For training the RFNNI in Fig.8, the cost function is defined as follows:

$$J_I(k)=\frac{1}{2}\sum_{s=1}^{p}\left(e_{Is}(k)\right)^2=\sum_{s=1}^{p}\left(y_s(k)-y_{Is}(k)\right)^2 \tag{41}$$

where $y_s(k)$ is the sth output of the plant, $y_{Is}(k)=O_s^4$ is the sth output of RFNNI, and $e_{Is}(k)$ is the error between $y_s(k)$ and $y_{Is}(k)$ for each discrete time k.

By using the back propagation (BP) algorithm, the weights of the RFNNI is adjusted such

that the cost function defined in (41) is minimized. The BP algorithm may be written briefly as:

$$\begin{aligned} W_I(k+1) &= W_I(k) + \Delta W_I(k) \\ &= W_I(k) + \eta_I\left(-\frac{\partial J_I(k)}{\partial W_I(k)}\right) \end{aligned} \tag{42}$$

where η_I represents the learning rate and W_I represents the tuning weights, in this case, which are w^4_{Isq}, a_{Iiq_i}, b_{Iiqi}, and w^1_{Ii}. Subscript I represents RFNNI.

According to the RFNNI structure (34)~(37), cost function (41) and BP algorithm (42), the update rules of RFNNI weights are

$$w^4_{Isq}(k+1) = w^4_{Isq}(k) - \eta_I^{w4}\frac{\partial J_I(k)}{\partial w^4_{Isq}(k)} \tag{43}$$

$$a_{Iiq_i}(k+1) = a_{Iiq_i}(k) - \eta_I^{a}\frac{\partial J_I(k)}{\partial a_{Iiq_i}(k)} \tag{44}$$

$$b_{Iiq_i}(k+1) = b_{Iiq_i}(k) - \eta_I^{b}\frac{\partial J_I(k)}{\partial b_{Iiq_i}(k)} \tag{45}$$

$$w^1_{Ii}(k+1) = w^1_{Ii}(k) - \eta_I^{w1}\frac{\partial J_I(k)}{\partial w^1_{Ii}(k)} \tag{46}$$

Where

$$\frac{\partial J_I(k)}{\partial w^4_{Isq}(k)} = -e_{Is}(k)\frac{O^3_{Iq}}{\sum_q O^3_{Iq}}$$

$$\frac{\partial J_I(k)}{\partial a_{Iiq_i}(k)} = -\sum_s e_{Is}(k)\cdot\frac{w^4_{Isq} - O^4_{Is}}{\sum_q O^3_{Iq}}\cdot O^3_{Iq}\cdot\frac{2\left(O^1_{Ii} - a_{Iiq_i}\right)}{\left(b_{Iiq_i}\right)^2}$$

$$\frac{\partial J_I(k)}{\partial b_{Iiq_i}(k)} = -\sum_s e_{Is}(k)\cdot\frac{w^4_{Isq} - O^4_{Is}}{\sum_q O^3_{Iq}}\cdot O^3_{Iq}\cdot\frac{2\left(O^1_{Ii} - a_{Iiq_i}\right)^2}{\left(b_{Iiq_i}\right)^3}$$

$$\frac{\partial J_I(k)}{\partial w^1_{Ii}(k)} = -\sum_q\sum_s e_{Is}(k)\cdot\frac{w^4_{Isq} - O^4_{Is}}{\sum_q O^3_{Iq}}\cdot O^3_{Iq}\cdot\frac{-2\left(O^1_{Ii} - a_{Iiq_i}\right)}{\left(b_{Iiq_i}\right)^2}\cdot O^1_{Ii}(k-1)$$

3.3.2 Learning algorithm for controller

For training RFNNC in Fig. 8, the cost function is defined as

$$J_C(k)=\frac{1}{2}\sum_{s=1}^{p}(e_s(k))^2=\sum_{s=1}^{p}(r_s(k)-y_s(k))^2 \tag{47}$$

where $r_s(k)$ is the sth desired output, $y_s(k)$ is the sth actual system output and $e_s(k)$ is the error between $r_s(k)$ and $y_s(k)$.

Then, the gradient of J_C is

$$\begin{aligned}\frac{\partial J_C}{\partial W_C}&=\sum_s\frac{\partial J_C}{\partial y_s}\cdot\frac{\partial y_s}{\partial W_C}\\&=\sum_s -e_s(k)\frac{\partial y_s(k)}{\partial u_o(k)}\cdot\frac{\partial u_o(k)}{\partial W_C},\\&=\sum_s -e_s(k)yu_{so}(k)\cdot\frac{\partial u_o(k)}{\partial W_C}\end{aligned} \tag{48}$$

where u_o is the oth control signal, which is also the oth output of RFNNC, and $yu_{so}(k)=\partial y_s(k)/\partial u_o(k)$ denotes the system sensitivity. Thus the parameters of the RFNNC can be adjusted by

$$\begin{aligned}W_C(k+1)&=W_C(k)+\Delta W_C(k)\\&=W_C(k)+\eta_C(-\frac{\partial J_C(k)}{\partial W_C(k)})\end{aligned} \tag{49}$$

Note that the convergence of the RFNNC cannot be guaranteed until $yu_{so}(k)$ is known. Obviously, the RFNNI can provide $yu_{so}(k)$ to RFNNC. Resume that the oth control signal is also the oth input of RFNNI, then $yu_{so}(k)$ can be calculated by

$$\begin{aligned}\frac{\partial y_s(k)}{\partial u_o(k)}&=\sum_q\frac{\partial O_{Is}^4}{\partial O_{Iq}^3}\cdot\frac{\partial O_{Iq}^3}{\partial O_{Ioq_o}^2}\cdot\frac{\partial O_{Ioq_o}^2}{\partial O_{Io}^1}\cdot\frac{\partial O_{Io}^1}{\partial u_o}\\&=\sum_q\frac{w_{Isq}^4-O_{Is}^4}{\sum_q O_{Iq}^3}\cdot O_{Iq}^3\cdot\frac{-2(O_{Io}^1-a_{Ioq_o})}{(b_{Ioq_o})^2}\end{aligned} \tag{50}$$

3.4 Stability analysis of the RFNN

Choosing an appropriate learning rate η is very important for the stability of RFNN. If the value of the learning rate η is small, convergence of the RFNN can be guaranteed, however,

the convergence speed may be very slow. On the other hand, choosing a large value for the learning rate can fasten the convergence speed, but the system may become unstable.

3.4.1 Stability analysis for identifier

For choosing the appropriate learning rate for RFNNI, discrete Lyapunov function is defined as

$$L_I(k) = J_I(k) = \frac{1}{2}\sum_s (e_{Is}(k))^2 \tag{51}$$

Thus the change of the Lyapunov function due to the training process is

$$\begin{aligned}
\Delta L_I(k) &= L_I(k+1) - L_I(k) \\
&= \frac{1}{2}\sum_s (e_{Is}(k+1))^2 - \frac{1}{2}\sum_s (e_{Is}(k))^2 \\
&= \frac{1}{2}\sum_s \left[(e_{Is}(k+1))^2 - (e_{Is}(k))^2\right] \\
&= \frac{1}{2}\sum_s \left[(e_{Is}(k+1) + e_{Is}(k)) \cdot (e_{Is}(k+1) - e_{Is}(k))\right] \\
&= \frac{1}{2}\sum_s \left[(2e_{Is}(k) + \Delta e_{Is}(k)) \cdot \Delta e_{Is}(k)\right] \\
&= \frac{1}{2}\sum_s \left[(\Delta e_{Is}(k))^2 + 2e_{Is}(k)\Delta e_{Is}(k)\right] \\
&= \frac{1}{2}\sum_s (\Delta e_{Is}(k))^2 + \frac{1}{2}\sum_s \left[2e_{Is}(k)\Delta e_{Is}(k)\right]
\end{aligned} \tag{52}$$

The error difference due to the learning can be represented by

$$\Delta e_{Is}(k) = e_{Is}(k+1) - e_{Is}(k) \approx \frac{\partial e_{Is}(k)}{\partial W_I(k)} \cdot \Delta W_I(k) \tag{53}$$

Where

$$\begin{aligned}
\Delta W_I(k) &= -\eta_I \frac{\partial J_I(k)}{\partial W_I(k)} = -\eta_I \sum_s \frac{\partial J_I(k)}{\partial e_{Is}(k)} \cdot \frac{\partial e_{Is}(k)}{\partial W_I(k)} \\
&= -\eta_I \sum_s e_{Is}(k) \cdot \frac{\partial e_{Is}(k)}{\partial W_I(k)}
\end{aligned}$$

So (52) can be modified as

$$\begin{aligned}\Delta L(k) &= \frac{1}{2}\sum_s\left[\frac{\partial e_{Is}(k)}{\partial W_I(k)}\cdot\left(-\eta_I\frac{\partial J_I(k)}{\partial W_I(k)}\right)\right]^2+\frac{1}{2}\sum_s\left[2e_{Is}(k)\cdot\frac{\partial e_{Is}(k)}{\partial W_I(k)}\cdot\left(-\eta_I\frac{\partial J_I(k)}{\partial W_I(k)}\right)\right]\\ &=\frac{1}{2}\left(\eta_I\frac{\partial J_I(k)}{\partial W_I(k)}\right)^2\sum_s\left(\frac{\partial e_{Is}(k)}{\partial W_I(k)}\right)^2-\eta_I\frac{\partial J_I(k)}{\partial W_I(k)}\cdot\sum_s\left(e_{Is}(k)\cdot\frac{\partial e_{Is}(k)}{\partial W_I(k)}\right)\\ &=\frac{1}{2}\left(\eta_I\frac{\partial J_I(k)}{\partial W_I(k)}\right)^2\sum_s\left(\frac{\partial e_{Is}(k)}{\partial W_I(k)}\right)^2-\eta_I\left(\frac{\partial J_I(k)}{\partial W_I(k)}\right)^2\\ &=\frac{1}{2}\eta_I\left(\frac{\partial J_I(k)}{\partial W_I(k)}\right)^2\left[\eta_I\sum_s\left(\frac{\partial e_{Is}(k)}{\partial W_I(k)}\right)^2-2\right]\end{aligned} \tag{54}$$

To guarantee the convergence of RFNNI, the change of Lyapunov function $\Delta L_I(k)$ should be negative. So learning rate must satisfy the following condition:

$$0<\eta_I(k)<2\Big/\sum_s\left(\frac{\partial e_{Is}(k)}{\partial W_I(k)}\right)^2 . \tag{55}$$

For the learning rate of each weight in RFNNI, the condition (22) can be modified as

$$0<\eta_I^{w4}(k)<2\Big/\max_q\left\{\sum_s\left(\frac{\partial e_{Is}(k)}{\partial w^4_{Isq}(k)}\right)^2\right\} \tag{56}$$

$$0<\eta_I^{a}(k)<2\Big/\max_{q,i}\left\{\sum_s\left(\frac{\partial e_{Is}(k)}{\partial a_{Iiq_i}(k)}\right)^2\right\} \tag{57}$$

$$0<\eta_I^{b}(k)<2\Big/\max_{q,i}\left\{\sum_s\left(\frac{\partial e_{Is}(k)}{\partial b_{Iiq_i}(k)}\right)^2\right\} \tag{58}$$

$$0<\eta_I^{w1}(k)<2\Big/\max_{i}\left\{\sum_s\left(\frac{\partial e_{Is}(k)}{\partial w^1_{Ii}(k)}\right)^2\right\} . \tag{59}$$

3.4.2 Stability analysis for controller

Similar to (51), the Lyapunov function for RFNNC can be defined as

$$L_C(k) = J_C(k) = \frac{1}{2}\sum_s (e_s(k))^2 \tag{60}$$

So, similar to (56)-(59), the learning rates for training RFNNC should be chosen according to the following rules:

$$0 < \eta_C^{w4}(k) < 2 \Big/ \max_q \left\{ \sum_s \left(\frac{\partial e_s(k)}{\partial w^4_{C\,sq}(k)} \right)^2 \right\} \tag{61}$$

$$0 < \eta_C^{a}(k) < 2 \Big/ \max_{q,i} \left\{ \sum_s \left(\frac{\partial e_s(k)}{\partial a_{C\,iq_i}(k)} \right)^2 \right\} \tag{62}$$

$$0 < \eta_C^{b}(k) < 2 \Big/ \max_{q,i} \left\{ \sum_s \left(\frac{\partial e_s(k)}{\partial b_{C\,iq_i}(k)} \right)^2 \right\} \tag{63}$$

$$0 < \eta_C^{w1}(k) < 2 \Big/ \max_i \left\{ \sum_s \left(\frac{\partial e_s(k)}{\partial w^1_{C\,i}(k)} \right)^2 \right\} \tag{64}$$

3.5 Simulation Experiments

Dynamics of robotic manipulators are highly nonlinear and may contain uncertain elements such as friction and load. Many efforts have been made in developing control schemes to achieve the precise tracking control of robot manipulators. Among available options, neural networks and fuzzy systems (Er & Chin 2000; Llama et al. 2000; Wang & Lin 2000; Huang & Lian 1997) are used more and more frequently in recent years. In the simulation experiments of this chapter, the proposed RFNNBAC is applied to control the trajectory of the two-link robotic manipulator described in chapter 2.4 to prove its effectiveness.

In the simulation, the parameters of manipulator are m_1 =4 *kg*, m_2 =2 kg, l_1 =1 *m*, l_2 =0.5 *m*, *g* =9.8 *N/kg*. Initial conditions are given as $\theta_1(0)$ =0 *rad*, $\theta_2(0)$ =1 *rad*, $\dot{\theta}_1(0)$ =0, and $\dot{\theta}_2(0)$ =0 *rad/s*. The desired trajectory is given by $\hat{\theta}_1(t) = \sin(2\pi t)$ and $\hat{\theta}_2(t) = \cos(2\pi t)$. The friction and disturbance terms in (4) are assumed to be

$$d_R = \begin{bmatrix} 5\cos(5t) \\ 5\cos(5t) \end{bmatrix} Nm, \quad \Delta T(q, \dot{q}) = 0.5\,\mathrm{sign}(\dot{q})\, Nm.$$

Simulation results are shown in Fig.9 ~Fig.14. Fig.9 and Fig.10 illustrate the trajectories of two joints; the two outputs of identifier (RFNNI) are shown in Fig.11 and Fig.12 separately; the cost function for RFNNC is shown in Fig.13; and Fig.14 shows the cost function for RFNNI.

From simulation results, it is obvious that the proposed RFNN can identify and control the robot manipulator very well.

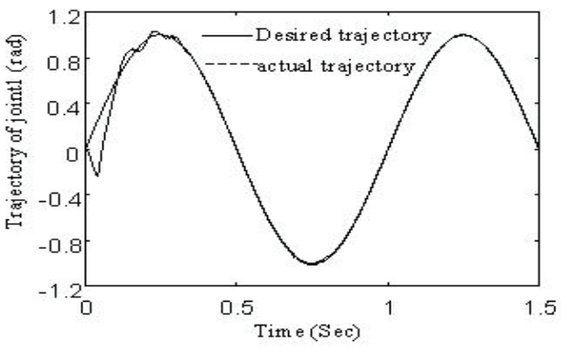

Fig. 9. Trajectory of joint1

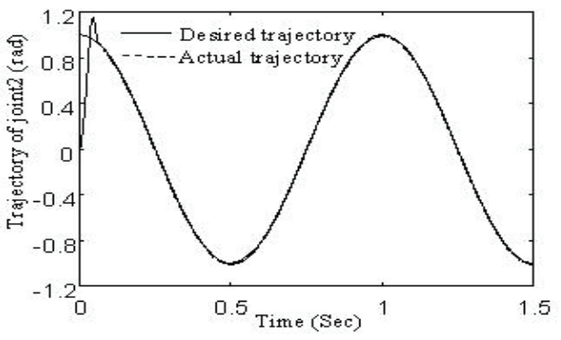

Fig. 10. Trajectory of joint2

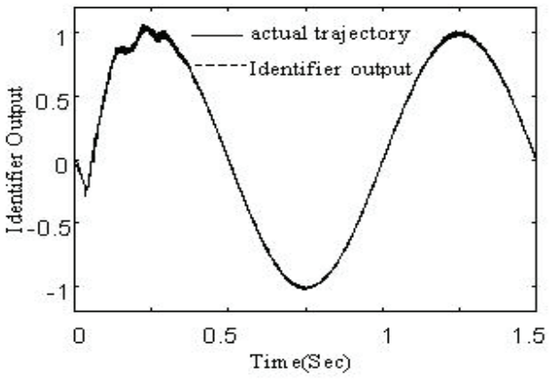

Fig. 11. Identifier (RFNNI) output1

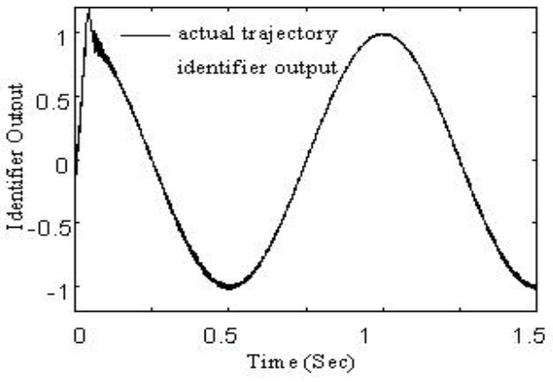

Fig. 12. Identifier (RFNNC) output2

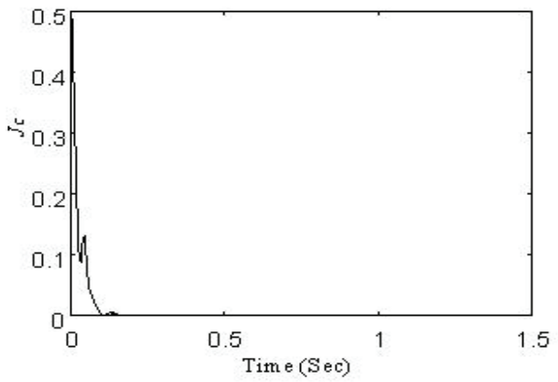

Fig. 13. Cost function for RFNNC

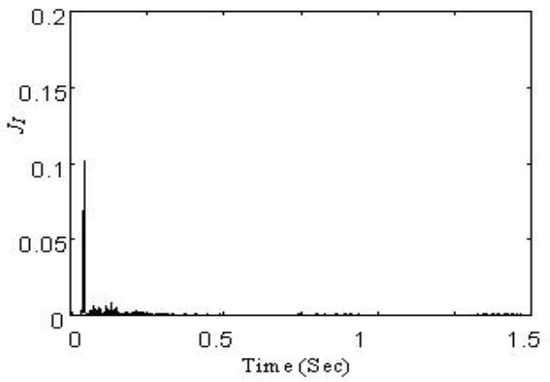

Fig. 14. Cost function for RFNNI

4. Conclusion

In this paper, the adaptive control based on neural network is studied. Firstly, a neural network based adaptive robust tracking control design is proposed for robotic systems under the existence of uncertainties. In this proposed control strategy, the NN is used to identify the modeling uncertainties, and then the disadvantageous effects caused by neural network approximating error and external disturbances in robotic system are counteracted by robust controller. Especially the proposed control strategy is designed based on HJI inequation theorem to overcome the approximation error of the neural network bounded issue. Simulation results show that proposed control strategy is effective and has better performance than traditional robust control strategy. Secondly, an RFNN for realizing fuzzy inference using the dynamic fuzzy rules is proposed. The proposed RFNN consists of four layers and the feedback connections are added in first layer. The proposed RFNN can be used for the identification and control of dynamic system. For identification, RFNN only needs the current inputs and most recent outputs of system as its inputs. For control, two RFNNs are used to constitute an adaptive control system, one is used as identifier (RFNNI) and another is used as controller (RFNNC). Also to prove the proposed RFNN and control strategy robust, it is used to control the robot manipulator and simulation results verified their effectiveness.

5. References

Abdallah, C., Dawson, D., Dorato, P. & Jamshidi, M. (1991). Survey of the robust of rigid robots, *IEEE Control Systems Magazine,* Vol. 11, No. 2, pp. 24-30.

Ortega, R. & Spong, M. W. (1989). Adaptive motion control of rigid robots: a tutorial, *Automatica,* Vol. 25, No. 3, pp. 877-888.

Saad, M., Dessaint, L. A., Bigras, P. & Haddad, K. (1994). Adaptive versus neural adaptive control: application to robotics, *International Journal of Adaptive Control and Signal Processing,* Vol. 8, No. 2, pp. 223-236.

Sanner, R. M. & Slotine, J. J. E. (1992). Gaussian networks for direct adaptive control, *IEEE Transactions on. Neural Network,* Vol. 3, No. 4, pp. 837-863.

Spooner, J. T. & Passino, K. M. (1996). Stable adaptive control using fuzzy systems and neural networks, *IEEE Transactions on Fuzzy system,* Vol. 4, No. 2, pp. 339-359.

Narenra, K. S. & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural networks,* Vol. 1, No. 1, pp. 4-27.

Polycarpou, M. M. (1996). Stable adaptive neural control scheme for nonlinear systems, *IEEE Transactions on Automatic Control,* Vol. 41, No. 2, pp. 447-451.

Carelli, R., Camacho, E. F. & Patino, D. (1995). A neural network based feedforward adaptive controller for robot, *IEEE Transactions on Systems, Mman and Cybernetics, Part B: Cybernetics,* Vol. 25, No. 6, pp. 1281-1288.

Behera, L., Chaudhury, S. & Gopal, M. (1996). Neuro-adaptive hybrid controller for robot-manipulator tracking control, *IEE Proceedings Control Theory Applications,* Vol.143, No.1, pp.2710-275.

Shen, T. L. (1996). *H∞ control theory and its applications*, ISBN 7302022151, Tsinghua Press, Beijin, China.

Park, Y. M., Choi, M. S. & Lee, K. Y. (1996). An optimal tracking neuro-controller for nonlinear dynamic systems, *IEEE Transactions on Neural Networks,* Vol. 7, No. 5, pp. 1099-1110.

Narendra, K. S. & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks,* Vol. 1, No. 1, pp. 4-27.

Brdys, M. A. & Kulawski, G. J. (1999). Dynamic neural controllers for induction motor, *IEEE Transactions on Neural Networks,* Vol. 10, No. 2, pp. 340-355.

Ku, C. C. & Lee, K. Y. (1995). Diagonal recurrent neural networks for dynamic systems control, *IEEE Transactions on Neural Networks,* Vol. 6, No. 1, pp. 144-156.

Ma, S. & Ji, C. (1998). Fast training of recurrent neural networks based on the EM algorithm, *IEEE Transactions on Neural Networks,* Vol. 9, No. 1, pp. 11-26.

Sundareshan, M. K. & Condarcure, T. A. (1998). Recurrent neural-network training by a learning automation approach for trajectory learning and control system design, *IEEE Transactions on Neural Networks,* Vol. 9, No. 3, pp. 354-368.

Liang, X. B. & Wang, J. (2000). A recurrent neural network for nonlinear optimization with a continuously differentiable objective function and bound constraints, *IEEE Transactions on Neural Networks,* Vol. 11, No. 6, pp. 1251-1262.

Jang, J. S. R. & Sun, C. T. (1993). Functional equivalence between radial basis function networks and fuzzy inference systems, *IEEE Transactions on Neural Networks*, Vol. 4, No. 1, pp. 156-159.

Hunt, K. J., Hass, R. & Munay-Smith, R. (1996). Extending the functional equivalence of radial basis function networks and fuzzy inference systems, *IEEE Transactions on Neural Networks*, Vol. 7, No. 3, pp. 776-781.

Buckley, J. J., Hayashi, Y. & Czogala, E. (1993). On the equivalence of neural nets and fuzzy expert systems, *Fuzzy Sets and Systems*, Vol. 53, No. 2, pp. 129-134.

Reyneri, L. M. (1999). Unification of neural and wavelet networks and fuzzy systems, *IEEE Transactions on Neural Networks*, Vol. 10, No. 4, pp. 801-814.

Er, M. J. & Chin, S. H. (2000). Hybrid adaptive fuzzy controller of robot manipulators with bounds estimation, *IEEE Transactions on Industrial Electronics*, Vol. 47, No. 5, pp. 1151-1160.

Llama, M. A., Kelly, R. & Santibanez, V. (2000). Stable computed-torque control of robot manipulator via fuzzy self-tuning, *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, Vol. 30, No. 1, pp. 143-150.

Wang, S. D. & Lin, C. K. (2000). Adaptive tuning of the fuzzy controller for robots, *Fuzzy Sets Systems,* Vol. 110, No. 3, pp. 351-363.

Huang, S. J. & Lian, R. J. (1997). A hybrid fuzzy logic and neural network algorithm for robot motion control, *IEEE Transactions on Industrial Electronics*, Vol. 44, No. 3, pp. 408-417.

Adaptive Control

Edited by Kwanho You

ISBN 978-953-7619-47-3
Hard cover, 372 pages
Publisher InTech
Published online 01, January, 2009
Published in print edition January, 2009

Adaptive control has been a remarkable field for industrial and academic research since 1950s. Since more and more adaptive algorithms are applied in various control applications, it is becoming very important for practical implementation. As it can be confirmed from the increasing number of conferences and journals on adaptive control topics, it is certain that the adaptive control is a significant guidance for technology development.The authors the chapters in this book are professionals in their areas and their recent research results are presented in this book which will also provide new ideas for improved performance of various control application problems.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Sun Wei, Zhang Lujin, Zou Jinhai and Miao Siyi (2009). Adaptive Control Based On Neural Network, Adaptive Control, Kwanho You (Ed.), ISBN: 978-953-7619-47-3, InTech, Available from: http://www.intechopen.com/books/adaptive_control/adaptive_control_based_on_neural_network

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2009 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen